

EXHIBIT B

Chapter 14

DIDS (Distributed Intrusion Detection System)—Motivation, Architecture, and an Early Prototype

*Steven R. Snapp, James Brentano, Giban V. Dias,
Terrance L. Goan, L. Todd Heberlein, Che-Lin Ho,
Karl N. Levitt, Biswanath Mukherjee, Stephen E. Smaha,
Tim Grance, Daniel M. Teal, and Doug Mansur*

Intrusion detection is the problem of identifying unauthorized use, misuse, and abuse of computer systems by both system insiders and external penetrators. The proliferation of heterogeneous computer networks provides additional implications for the intrusion detection problem. Namely, the increased connectivity of computer systems gives greater access to outsiders, and makes it easier for intruders to avoid detection. IDSs are based on the belief that an intruder's behavior will be noticeably different from that of a legitimate user. We are designing and implementing a prototype Distributed Intrusion Detection System (DIDS) that combines distributed monitoring and data reduction (through individual host and LAN monitors) with centralized data analysis (through the DIDS director) to monitor a heterogeneous network of computers. This approach is unique among current IDSs. A main problem considered in this chapter is the network-user identification problem, which is concerned with tracking a user moving across the network, possibly with a new user-ID on each computer. Initial system prototypes have provided quite favorable results on this problem and the detection of attacks on a network. This chapter provides an overview of the motivation behind DIDS, the system architecture and capabilities, and a discussion of the early prototype.

Proceedings of the 14th National Computer Security Conference, October 1991.

212 DIDS (Distributed Intrusion Detection System)

INTRODUCTION

Intrusion detection is defined to be the problem of identifying individuals who are using a computer system without authorization (i.e., *crackers*) and those who have legitimate access to the system but are exceeding their privileges (i.e., the *inside threat*). Work is being done elsewhere on Intrusion Detection Systems (IDSs) for a single host [8, 10, 11] and for several hosts connected by a network [6, 7, 12]. Our own earlier work on the Network Security Monitor (NSM) concentrated on monitoring a broadcast Local Area Network (LAN) [3].

The proliferation of heterogeneous computer networks has serious implications for the intrusion detection problem. Foremost among these implications is the increased opportunity for unauthorized access that is provided by the network's connectivity. This problem is exacerbated when dial-up or internet-work access is allowed, as well as when unmonitored hosts (hosts without audit trails) are present. The use of distributed rather than centralized computing resources also implies reduced control over those resources. Moreover, multiple independent computers are likely to generate more audit data than a single computer, and this audit data is dispersed among the various systems. Clearly, not all of the audit data can be forwarded to a single IDS for analysis; some analysis must be accomplished locally.

This chapter describes a prototype Distributed Intrusion Detection System (DIDS) which generalizes the target environment in order to monitor multiple hosts connected via a network as well as the network itself. The DIDS components include the DIDS director, a single host monitor per host, and a single LAN monitor for each LAN segment of the monitored network. The information gathered by these distributed components is transported to, and analyzed at, a central location (i.e., an expert system, which is a sub-component of the director), thus providing the capability to aggregate information from different sources. We can cope with any audit trail format as long as the events of interest are provided.

DIDS is designed to operate in a heterogeneous environment composed of C2 [1] or higher-rated computers. The current target environment consists of several hosts connected by a broadcast LAN segment (presently an Ethernet; see Figure 1). The use of C2-rated systems implies a consistency in the content of the system audit trails. This allows us to develop standard representations into which we can map audit data from UNIX, VMS, or any other system with C2 auditing capabilities. The C2 rating also guarantees, as part of the Trusted Computing Base (TCB), the security and integrity of the host's audit records. Although the hosts must comply with the C2 specifications in order to be monitored directly, the network-related activity of non-compliant hosts can be monitored via the LAN monitor. Since all attacks that utilize the network for system access will

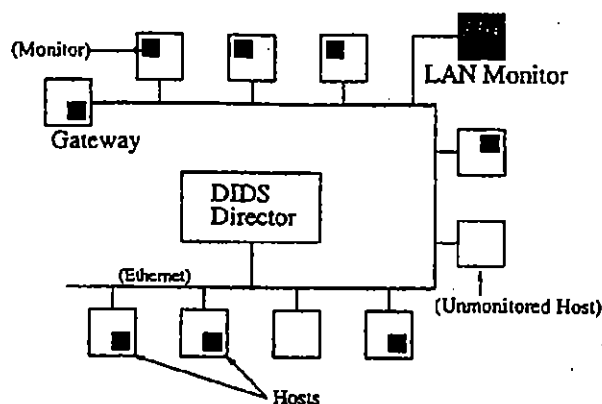


FIGURE 1 DIDS target environment.

pass through the LAN segment, the LAN monitor will be able to monitor all of this traffic.

The next section describes the type of behavior that DIDS is intended to detect. Following this section we present an overview of the DIDS architecture; formulate the concept of the network-user identification (NID), an identifier for a network-wide user, and describe its use in distributed intrusion detection; give an overview of the host and LAN monitors, respectively; and discuss the expert system and its processing mechanisms based on the NID. The last section provides some concluding remarks.

SCENARIOS

The detection of certain attacks against a networked system of computers requires information from multiple sources. A simple example of such an attack is the so-called *doorknob* attack. In a doorknob attack the intruder's goal is to discover, and gain access to, insufficiently protected hosts on a system. The intruder generally tries a few common account and password combinations on each of a number of computers. These simple attacks can be remarkably successful [4]. As a case in point, UC Davis' NSM recently observed an attacker of this type gaining super-user access to an external computer that did not require a password for the super-user account. In this case, the intruder used *telnet* to make the connection from a university computer system, and then repeatedly tried to gain access to several different computers at the external site. In cases like these, the intruder tries only a few logins on each machine (usually with different account names), which means that an IDS on each host may

214 DIDS (Distributed Intrusion Detection System)

not flag the attack. Even if the behavior is recognized as an attack on the individual host, current IDSs are generally unable to correlate reports from multiple hosts; thus they cannot recognize the *doorknob* attack as such. Because DIDS aggregates and correlates data from multiple hosts and the network, it is in a position to recognize the doorknob attack by detecting the pattern of repeated failed logins, even though there may be too few on a single host to alert that host's monitor.

In another incident, our NSM recently observed an intruder gaining access to a computer using a guest account that did not require a password. Once the attacker had access to the system, he exhibited behavior which would have alerted most existing IDSs (e.g., changing passwords and failed events). In an incident such as this, DIDS would not only report the attack, but may also be able to identify the source of the attack. That is, while most IDSs would report the occurrence of an incident involving user "guest" on the target machine, DIDS would also report that user "guest" was really, for example, user "smith" on the source machine, assuming that the source machine was in the monitored domain. It may also be possible to go even further back and identify all of the different user accounts in the "chain" to find the initial launching point of the attack.

Another possible scenario is what we call *network browsing*. This occurs when a (network) user is looking through a number of files on several different computers within a short period of time. The browsing activity level on any single host may not be high enough to raise any alarm by itself. However, the network-wide, aggregated browsing activity level may be high enough to raise suspicion on this user. Network browsing can be detected as follows. Each host monitor will report that a particular user is browsing on that system, even if the corresponding degree of browsing is small. The expert system can then aggregate such information from multiple hosts to determine if all of the browsing activity corresponds to the same network user. This scenario presents a key challenge for DIDS: the tradeoff between sending all audit records to the director versus missing attacks because thresholds on each host are not exceeded.

In addition to the specific scenarios outlined above, there are a number of general ways that an intruder can use the connectivity of the network to hide his trail and to enhance his effectiveness. Some of the attack configurations that have been hypothesized include *chain* and *parallel* attacks [2]. DIDS combats these inherent vulnerabilities of the network by using the very same connectivity to help track and detect the intruder. Note that DIDS should be at least as effective as host-based IDSs (if we implement all of their functionality in the DIDS host monitor), and at least as effective as the stand-alone NSM.

DIDS ARCHITECTURE

The DIDS architecture combines distributed monitoring and data reduction with centralized data analysis. This approach is unique among current IDSs. The components of DIDS are the *DIDS director*, a single *host monitor* per host, and a single *LAN monitor* for each broadcast LAN segment in the monitored network. DIDS can potentially handle hosts without monitors since the LAN monitor can report on the network activities of such hosts. The host and LAN monitors are primarily responsible for the collection of evidence of unauthorized or suspicious activity, while the DIDS director is primarily responsible for its evaluation. Reports are sent independently and asynchronously from the host and LAN monitors to the DIDS director through a communications infrastructure (Figure 2). High-level communication protocols between the components are based on the ISO Common Management Information Protocol (CMIP) recommendations, allowing for future inclusion of CMIP management tools as they become useful. The architecture also provides for bidirectional communication between the DIDS director and any monitor in the configuration. This communication consists primarily of notable events and anomaly reports from the monitors. The director can also make requests for more detailed information from the distributed monitors via a "GET" directive, and issue commands to have the distributed monitors modify their monitoring capabilities via a "SET" directive. A large amount of low-level filtering and some analysis are performed by the host monitor to minimize the use of network bandwidth in passing evidence to the director.

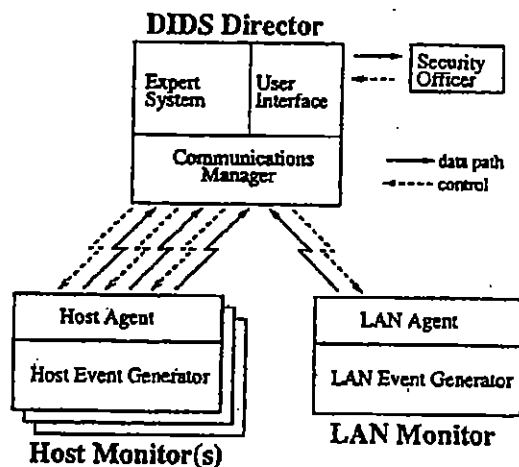


FIGURE 2 Communications architecture.

216 DIDS (Distributed Intrusion Detection System)

The host monitor consists of a *host event generator* (HEG) and a *host agent*. The HEG collects and analyzes audit records from the host's operating system. The audit records are scanned for *notable events*, which are transactions that are of interest independent of any other records. These include, among others, failed events, user authentications, changes to the security state of the system, and any network access such as *rlogin* and *rsh*. These notable events are then sent to the director for further analysis. In enhancements under development, the HEG will also track user sessions and report anomalous behavior aggregated over time through user/group profiles and the integration of Haystack [10] into DIDS. The host agent handles all communications between the host monitor and the DIDS director.

Like the host monitor, the LAN monitor consists of a *LAN event generator* (LEG) and a *LAN agent*. The LEG is currently a subset of UC Davis' NSM [3]. Its main responsibility is to observe all of the traffic on its segment of the LAN to monitor host-to-host connections, services used, and volume of traffic. The LAN monitor reports on such network activity as *rlogin* and *telnet* connections, the use of security-related services, and changes in network traffic patterns.

The DIDS director consists of three major components that are all located on the same dedicated workstation. Because the components are logically independent processes, they could be distributed as well. The *communications manager* is responsible for the transfer of data between the director and each of the host and LAN monitors. It accepts the notable event records from each of the host and LAN monitors and sends them to the *expert system*. On behalf of the expert system or user interface, it is also able to send requests to the host and LAN monitors for more information regarding a particular subject. The expert system is responsible for evaluating and reporting on the security state of the monitored system. It receives the reports from the host and LAN monitors and, based on these reports, makes inferences about the security of each individual host, as well as the system as a whole. The expert system is a rule-based system with simple learning capabilities. The director's *user interface* allows the System Security Officer (SSO) interactive access to the entire system. The SSO is able to watch activities on each host, watch network traffic (by setting "wire-taps"), and request more specific types of information from the monitors.

We anticipate that a growing set of tools, including incident-handling tools and network-management tools, will be used in conjunction with the intrusion-detection functions of DIDS. This will give the SSO the ability to actively respond to attacks against the system in real-time. Incident-handling tools may consist of possible courses of action to take against an attacker, such as cutting off network access, a directed investigation of a particular user, removal of system access, etc. Network-management tools that are able to perform network mapping would also be useful.

THE NETWORK-USER IDENTIFICATION (NID)

One of the more interesting challenges for intrusion detection in a networked environment is to track users and objects (e.g., files) as they move across the network. For example, an intruder may use several different accounts on different machines during the course of an attack. Correlating data from several independent sources, including the network itself, can aid in recognizing this type of behavior and tracking an intruder to his source. In a networked environment, an intruder may often choose to employ the interconnectivity of the computers to hide his true identity and location. It may be that a single intruder uses multiple accounts to launch an attack, and that the behavior can be recognized as suspicious only if one knows that all of the activity emanates from a single source. For example, it is not particularly noteworthy if a user inquires about who is using a particular computer (e.g., using the UNIX *who* or *finger* command). However, it may be indicative of an attack if a user inquires about who is using each of the computers on a LAN and then subsequently logs into one of the hosts. Detecting this type of behavior requires attributing multiple sessions, perhaps with different account names, to a single source.

This problem is unique to the network environment and has not been dealt with before in this context. Our solution to the multiple-user identity problem is to create a *network-user identification* (NID) the first time a user enters the monitored environment, and then to apply that NID to any further instances of the user. All evidence about the behavior of any instance of the user is then accountable to the single NID. In particular, we must be able to determine that "smith@host1" is the same user as "jones@host2", if in fact they are. Since the network-user identification problem involves the collection and evaluation of data from both the host and LAN monitors, examining it is a useful method to understand the operation of DIDS. We will examine each of the components of DIDS in the context of the creation and use of the NID.

THE HOST MONITOR

The host monitor is currently installed on Sun SPARCstations running SunOS 4.0.x with the Sun C2 security package [9]. Through the C2 security package, the operating system produces audit records for virtually every transaction on the system. These transactions include file accesses, system calls, process executions, and logins. The contents of the Sun C2 audit record are: record type, record event, time, real user ID, audit user ID, effective user ID, real group ID, process ID, error code, return value, and label.

218 DIDS (Distributed Intrusion Detection System)

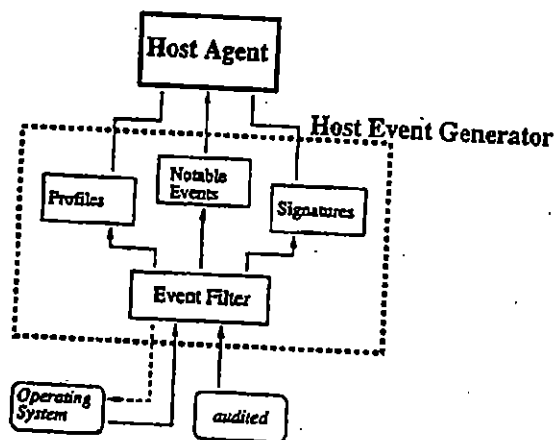


FIGURE 3 Host monitor structure.

The host monitor (Figure 3) examines each audit record to determine if it should be forwarded to the expert system for further evaluation. Certain critical audit records are always passed directly to the expert system (i.e., *notable events*); others are processed locally by the host monitor (i.e., *profiles* and attack *signatures*, which are sequences of noteworthy events that indicate the symptoms of attack) and only summary reports are sent to the expert system. Thus, one of the design objectives is to push as much of the processing operations down to the low-level monitors as possible. In order to do this, the HEG creates a more abstract object called an *event*. The event includes any significant data provided by the original audit record plus two new fields: the *action* and the *domain*. The action and domain are abstractions which are used to minimize operating system dependencies at higher levels. Actions characterize the dynamic aspect of the audit records. Domains characterize the objects of the audit records. In most cases, the objects are files or devices and their domain is determined by the characteristics of the object or its location in the file system. Since processes can also be objects of an audit record, they are also assigned to domains, in this case by their function.

The actions are: *session_start*, *session_end*, *read* (a file or device), *write* (a file or device), *execute* (a process), *terminate* (a process), *create* (a file or [virtual] device), *delete* (a file or [virtual] device), *move* (rename a file or device), *change_rights*, and *change_user_id*. The domains are: *tagged*, *authentication*, *audit*, *network*, *system*, *sys.info*, *user.info*, *utility*, *owned*, and *not_owned*.

The domains are prioritized so that an object is assigned to the first applicable domain. *Tagged* objects are ones that are thought a priori to be particularly interesting in terms of detecting intrusions. Any file, device, or process can be tagged (e.g., */etc/passwd*). *Authentication* objects are the processes

and files that are used to provide access control on the system (e.g., the password file). Similarly, *audit* objects relate to the accounting and security auditing processes and files. *Network* objects are the processes and files not covered in the previous domains which relate to the use of the network. *System* objects are primarily those that are concerned with the execution of the operating system itself, again exclusive of those objects already assigned to previously considered domains. *Sys_info* and *user_info* objects provide information about the system and about the users of the system, respectively. The *utility* objects are the bulk of the programs run by the users (e.g., compilers and editors). In general, the execution of an object in the utility domain is not interesting (except when the use is excessive), but the creation or modification of one is. *Owned* objects are relative to the user. *Not_owned* objects are, by exclusion, every object not assigned to a previous domain. They are also relative to a user; thus, files in the owned domain relative to "smith" are in the not_owned domain relative to "jones."

All possible transactions fall into one of a finite number of events formed by the cross product of the actions and the domains, and each event may also succeed or fail. Note that no distinction is made among files, directories, or devices, and that all of these are treated simply as objects. Not every action is applicable to every object; for example, the *terminate* action is applicable only to processes. The choice of these domains and actions is somewhat arbitrary in that one could easily suggest both finer- and coarser-grained partitions. However, they capture most of the interesting behavior for intrusion detection and correspond reasonably well with what other researchers in this field have found to be of interest [5, 10]. By mapping an infinite number of transactions to a finite number of events, we not only remove operating system dependencies, but also restrict the number of permutations with which the expert system will have to deal. The concept of the domain is one of the keys to detecting abuses. Using the domain allows us to make assertions about the nature of a user's behavior in a straightforward and systematic way. Although we lose some details provided by the raw audit information, it is more than made up for by the increase in portability, speed, simplicity, and generality.

An event reported by a host monitor is called a host audit record (*har*). The record syntax is: *har*(Monitor-ID, Host-ID, Audit-UID, Real-UID, Effective-UID, Time, Domain, Action, Transaction, Object, Parent Process, PID, Return Value, Error Code).

Of all the possible events, only a subset is forwarded to the expert system. For the creation and application of the NID, it is the events that relate to the creation of user sessions or to a change in an account that are important. These include all the events with *session_start* actions, as well as ones with an *execute* action applied to the *network* domain. These latter events capture such transactions as executing the *rlogin*, *telnet*, *rsh*, and *rexec* UNIX programs. The HEG consults external tables, which are built by hand, to determine which

220 DIDS (Distributed Intrusion Detection System)

events should be forwarded to the expert system. Because they relate to events rather than to the audit records themselves, the tables and the modules of the HEG which use them are portable across operating systems. The only portion of the HEG that is operating-system dependent is the module that creates the events.

THE LAN MONITOR

The LAN monitor is currently a subset of UC Davis' Network Security Monitor [3]. The LAN monitor builds its own "LAN audit trail." The LAN monitor observes each and every packet on its segment of the LAN and, from these packets, it is able to construct higher-level objects such as connections (logical circuits), and service requests using the TCP/IP or UDP/IP protocols. In particular, it audits host-to-host connections, services used, and volume of traffic per connection.

Similar to the host monitor, the LAN monitor uses several simple analysis techniques to identify significant events. The events include the use of certain services (e.g., *rlogin* and *telnet*) as well as activity by certain classes of hosts (e.g., a PC without a host monitor). The LAN monitor also uses and maintains profiles of expected network behavior. The profiles consist of expected data paths (e.g., which systems are expected to establish communication paths to which other systems, and by which service) and service profiles (e.g., what a typical *telnet*, *mail*, or *finger* is expected to look like).

The LAN monitor also uses heuristics in an attempt to identify the likelihood that a particular connection represents intrusive behavior. These heuristics consider the capabilities of each of the network services, the level of authentication required for each of the services, the security level for each machine on the network, and signatures of past attacks. The abnormality of a connection is based on the probability of that particular connection occurring and the behavior of the connection itself. Upon request, the LAN monitor is also able to provide a more detailed examination of any connection, including capturing every character crossing the network (i.e., a wire-tap). This capability can be used to support a directed investigation of a particular subject or object. Like the host monitor, the LAN monitor forwards relevant security information to the director through its LAN agent.

An event reported by a LAN monitor is called a network audit record (*nar*). The record syntax is: *nar*(Monitor-ID, Source_Host, Dest_Host, Time, Service, Domain, Status).

The LAN monitor has several responsibilities with respect to the creation and use of the NID. The LAN monitor is responsible for detecting any connec-

tions related to *rlogin* and *telnet* sessions. Once these connections are detected, the LAN monitor can be used to verify the owner of a connection. The LAN monitor can also be used to help track tagged objects moving across the network. The SSO can also ask for a wire-tap on a certain network connection to monitor a particular user's behavior.

THE EXPERT SYSTEM

DIDS utilizes a rule-based (or production) expert system. The expert system is currently written in Prolog, and much of the form of the rule base comes from Prolog and the logic notation that Prolog implies. The expert system uses rules derived from the hierarchical Intrusion Detection Model (IDM). The IDM describes the data abstractions used in inferring an attack on a network of computers. That is, it describes the transformation from the distributed raw audit data to high-level hypotheses about intrusions and about the overall security of the monitored environment. In abstracting and correlating data from the distributed sources, the model builds a virtual machine which consists of all the connected hosts as well as the network itself. This unified view of the distributed system simplifies the recognition of intrusion behavior which spans individual hosts. The model is also applicable to the trivial network of a single computer.

The model is the basis of the rule base. It serves both as a description of the function of the rule base and as a touchstone for the actual development of the rules. The IDM consists of six layers, each layer representing the result of a transformation performed on the data (see Table 1).

TABLE 1 Intrusion detection model.

Level	Name	Explanation
6	Security State	overall network security level
5	Threat	definition of categories of abuse
4	Context	event placed in context
3	Subject	definition and disambiguation of network user
2	Event	OS independent representation of user action (finite number of these)
1	Data	audit or OS provided data

222 DIDS (Distributed Intrusion Detection System)

The objects at the first level of the model are the audit records provided by the host operating system, by the LAN monitor, or by a third-party auditing package. The objects at this level are both syntactically and semantically dependent on the source. At this level, all of the activity on the host or LAN is represented.

At the second level, the *event* (which has already been discussed in the context of the host and LAN monitors) is both syntactically and semantically independent of the source standard format for events.

The third layer of the IDM creates a *subject*. This introduces a single identification for a user across many hosts on the network. It is the subject who is identified by the NID (see the next section). Upper layers of the model treat the network user as a single entity, essentially ignoring the local identification on each host. Similarly, above this level, the collection of hosts on the LAN are generally treated as a single distributed system with little attention being paid to the individual hosts.

The fourth layer of the model introduces the event in *context*. There are two kinds of context: temporal and spatial. As an example of temporal context, behavior that is unremarkable during standard working hours may be highly suspicious during off-hours [5]. The IDM, therefore, allows for the application of information about wall-clock time to the events it is considering. Wall-clock time refers to information about the time of day, weekdays versus weekends and holidays, as well as periods when an increase in activity is expected. In addition to the consideration of external temporal context, the expert system uses time windows to correlate events occurring in temporal proximity. This notion of temporal proximity implements the heuristic that a call to the UNIX *who* command followed closely by a *login* or *logout* is more likely to be related to an intrusion than either of those events occurring alone. Spatial context implies the relative importance of the source of events. That is, events related to a particular user, or events from a particular host, may be more likely to represent an intrusion than similar events from a different source. For example, a user moving from a low-security machine to a high-security machine may be of greater concern than a user moving in the opposite direction. The model also allows for the correlation of multiple events from the same user or source. In both of these cases, multiple events are more noteworthy when they have a common element than when they do not.

The fifth layer of the model considers the *threats* to the network and the hosts connected to it. Events in context are combined to create threats. The threats are partitioned by the nature of the abuse and the nature of the target. In other words, what is the intruder doing, and what is he doing it to? Abuses are divided into *attacks*, *misuses*, and *suspicious acts*. Attacks represent abuses in which the state of the machine is changed. That is, the file system or process state is different after the attack than it was prior to the attack. Misuses represent out-

of-policy behavior in which the state of the machine is not affected. Suspicious acts are events which, while not a violation of policy, are of interest to an IDS. For example, commands that provide information about the state of the system may be suspicious. The targets of abuse are characterized as being either *system* objects or *user* objects and as being either *passive* or *active*. User objects are owned by non-privileged users and/or reside within a non-privileged user's directory hierarchy. System objects are the complement of user objects. Passive objects are files, including executable binaries, while active objects are essentially running processes.

At the highest level, the model produces a numeric value between 1 and 100 that represents the overall *security state* of the network: the higher the number, the less secure the network. This value is a function of all the threats for all the subjects on the system. Here again we treat the collection of hosts as a single distributed system. Although representing the security level of the system as a single value seems to imply some loss of information, it provides a quick reference point for the SSO. In fact, in the current implementation, no information is lost since the expert system maintains all the evidence used in calculating the security state in its internal database, and the SSO has access to that database.

In the context of the network-user identification problem we are concerned primarily with the lowest three levels of the model: the audit data, the event, and the subject. The generation of the first two of these has already been discussed; thus, the creation of the subject is the focus of the following subsection.

The expert system is responsible for applying the rules to the evidence provided by the monitors. In general, the rules do not change during the execution of the expert system. What does change is a numerical value associated with each rule. This *Rule Value* (RV) represents our confidence that the rule is useful in detecting intrusions. These rule values are manipulated using a negative reinforcement training method that allows the expert system to continually lower the number of false attack reports. When a potential attack is reported by the expert system, the SSO determines the validity of the report and gives feedback to the expert system. If the report was deemed faulty, then the expert system lowers the RVs associated with the rules that were used to draw that conclusion. In addition to this directed training, which may lower some rule values, the system also automatically increases the RVs of all the rules on a regular basis. This recovery algorithm allows the system to adapt to changes in the environment as well as recover from faulty training.

Logically the rules have the form:

antecedent \Rightarrow consequence

where the antecedent is either a fact reported by one of the distributed monitors, or a consequence of some previously satisfied rule. The antecedent may also be

224 DIDS (Distributed Intrusion Detection System)

a conjunction of these. The overall structure of the rule base is a tree rooted at the top. Thus, many facts at the bottom of the tree will lead to a few conclusions at the top of the tree.

The expert system shell consists of approximately a hundred lines of Prolog source code. The shell is responsible for reading new facts reported by the distributed monitors, attempting to apply the rules to the facts and hypotheses in the Prolog database, reporting suspected intrusions, and maintaining the various dynamic values associated with the rules and hypotheses. The syntax for rules is:

$$\text{rule}(n, r, (\text{single}, [A]), (C)).$$

where n is the rule number, r is the initial RV, A is the single antecedent, and C is the consequence. Conjunctive rules have the form:

$$\text{rule}(n, r, (\text{and}, [A_1, A_2, A_3]), (C)).$$

where A_1 , A_2 , and A_3 are the antecedents and C is the consequence. Disjunctive rules are not allowed; this situation is dealt with by having multiple rules with the same consequence.

Building the NID

With respect to UNIX, the only legitimate ways to create an instance of a user are for the user to log in from a terminal, console, or off-LAN source, to change the user-ID in an existing instance, or to create additional instances (local or remote) from an existing instance. In each case, there is only one initial login (system wide) from an external device. When this original login is detected, a new unique NID is created. This NID is applied to every subsequent action generated by that user. When a user with a NID creates a new login session, that new session is associated with his original NID. Thus the system maintains a single identification for each physical user.

We consider an instance of a user to be the 4-tuple $\langle \text{session_start}, \text{user-id}, \text{host-id}, \text{time} \rangle$. Thus each login creates a new instance of a user. In associating an NID with an instance of a user, the expert system first tries to use an existing NID. If no NID can be found that applies to the instance, a new one is created. Trying to find an applicable existing NID consists of several steps. If a user changes identity (e.g., using UNIX's *su* command) on a host, the new instance is assigned the same NID as the previous identity. If a user performs a remote login from one host to another host, the new instance gets the same NID as the source instance. When no applicable NID is found, a new unique NID is created by the following rule:


```

rule(111,1000,[
    char(Host1,AUID,Time1,session_start,'local'), /* login */
    V+((ih(net_user(NID,AUID,Host,-)), /* no NID yet */
    newNID(X) /* create new NID */

    (net_user(X,AUID,Host1,Time1))). /* new net user */

```

The actual association of an NID with a user instance is through the hypothesis *net_user*. A new hypothesis is created for every event reported by the distributed monitors. This new hypothesis, called a *subject*, is formed by the rule:

```

rule(110,100,(and,[
    char(Mon,Host,AUID,UID,EUID,Time,Dom,Act,Trans,Obj,Parent,PID,
    Ret,Err).
    net_user(NID,AUID,Host,-)

    subj(NID,Mon,Host,AUID,UID,EUID,Time,Dom,Act,Trans,Obj,
    Parent,PID,Ret,Err))).

```

The rule creates a subject, getting the NID from the *net_user* and the remaining fields from the host audit record, if and only if both the user-ID and the host-ID match. It is through the use of the subject that the expert system correlates a user's actions regardless of the login name or host-ID.

There is still some uncertainty involved with the network-user identification problem. If a user leaves the monitored domain and then comes back in with a different user-ID, it is not possible to connect the two instances. Similarly, if a user passes through an unmonitored host, there is still uncertainty that any connection leaving the host is attributable to any connection entering the host. Multiple connections originating from the same host at approximately the same time also allow uncertainty if the user names do not provide any helpful information. The expert system can make a final decision with additional information from the host and LAN monitors that can (with high probability) disambiguate the connections.

CONCLUSIONS

Our Distributed Intrusion Detection System (DIDS) is being developed to address the shortcomings of current single host IDSs by generalizing the target environment to multiple hosts connected via a network (LAN). Most current IDSs do not consider the impact of the LAN structure when attempting to monitor user behavior for attacks against the system. Intrusion detection systems designed for a network environment will become increasingly important

226 DIDS (Distributed Intrusion Detection System)

as the number and size of LANs increase. Our prototype has demonstrated the viability of our distributed architecture in solving the network-user identification problem. We have tested the system on a sub-network of Sun SPARCstations and it has correctly identified network users in a variety of scenarios. Work continues on the design, development, and refinement of rules, particularly those which can take advantage of knowledge about particular kinds of attacks. The initial prototype expert system has been written in Prolog, but it is currently being ported to CLIPS due to the latter's superior performance characteristics and easy integration with the C programming language. We are designing a signature analysis component for the host monitor to detect events and sequences of events that are known to be indicative of an attack, based on a specific context. In addition to the current host monitor, which is designed to detect attacks on general-purpose multi-user computers, we intend to develop monitors for application-specific hosts such as file servers and gateways. In support of the ongoing development of DIDS we are planning to extend our model to a hierarchical Wide Area Network environment.

ACKNOWLEDGMENTS

The DIDS project is sponsored by the United States Air Force Cryptologic Support Center through a contract with the Lawrence Livermore National Labs.

REFERENCES

1. Department of Defense, *Trusted Computer System Evaluation Criteria*, National Computer Security Center, DOD 5200.28-STD, Dec. 1985.
2. G. V. Dias, K. N. Levitt, and B. Mukherjee, "Modeling Attacks on Computer Systems: Evaluating Vulnerabilities and Forming a Basis for Attack Detection," Technical Report CSE-90-41, University of California, Davis, July 1990.
3. L. T. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A Network Security Monitor," *Proceedings of the 1990 Symposium on Research in Security and Privacy*, pp. 296-304, Oakland, CA, May 1990.
4. B. Landreth, *Out of the Inner Circle, A Hacker's Guide to Computer Security*, Microsoft Press, Bellevue, WA, 1985.
5. T. Lunt, "Automated Audit Trail Analysis and Intrusion Detection: A Survey," *Proceedings of the 11th National Computer Security Conference*, pp. 65-73, Baltimore, MD, Oct. 1988.

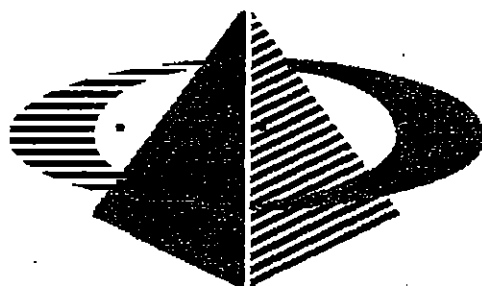
Steven R. Snapp et al. 227

6. T. E. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. G. Neumann, and C. Jalali, "IDES: A Progress Report," *Proceedings of the Sixth Annual Computer Security Applications Conference*, Tucson, AZ, Dec. 1990.
7. T. E. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, H. S. Javitz, A. Valdes, and P. G. Neumann, "A Real-Time Intrusion-Detection Expert System (IDES)," Interim Progress Report, Project 6784, SRI International, May 1990.
8. M. M. Sebring, E. Shellhouse, M. E. Hanna, and R. A. Whitehurst, "Expert Systems in Intrusion Detection: A Case Study," *Proceedings of the 11th National Computer Security Conference*, pp. 74-81, Baltimore, MD, Oct. 1988.
9. W. O. Sibert, "Auditing in a Distributed System: SunOS MLS Audit Trails," *Proceedings of the 11th National Computer Security Conference*, Baltimore, MD, Oct. 1988.
10. S. E. Smaha, "Haystack: An Intrusion Detection System," *Proceedings of the IEEE Fourth Aerospace Computer Security Applications Conference*, Orlando, FL, Dec. 1988.
11. H. S. Vaccaro and G. E. Liepins, "Detection of Anomalous Computer Session Activity," *Proceedings of the 1989 Symposium on Research in Security and Privacy*, pp. 280-289, Oakland, CA, May 1989.
12. J. R. Winkler, "A Unix Prototype for Intrusion and Anomaly Detection in Secure Networks," *Proceedings of the 13th National Computer Security Conference*, pp. 115-124, Washington, DC, Oct. 1990.

EXHIBIT C

WheelGroup Corporation

NetRanger High-Level Overview



Bob Gleichauf
VP of Engineering
bobg@wheelgroup.com
Dan Teal
Chief Scientist
teal@wheelgroup.com

Version 1.1

NETRANGER High-Level Overview

WheelGroup Corp
November 1996

NetRanger is a real-time network security management system that detects, analyzes, responds to, and deters unauthorized network activity. The NetRanger architecture supports large-scale information protection via centralized monitoring and management of remote dynamic packet filtering devices that plug into customer networks. Communication is maintained via WheelGroup Corporation's (WGC) proprietary secure communications architecture. Network activity can also be logged for more in-depth analysis.

As shown in Figure 1, NetRanger includes the following core systems and subsystems:

- **Network Security eXchange (NSX)**
 - Packet Filtering Device(s)
 - Sensor
- **Communication System**
 - Post Office(s)
 - Encrypted Sleeve
- **Director**
 - Security Management Interface
 - Security Analysis Package

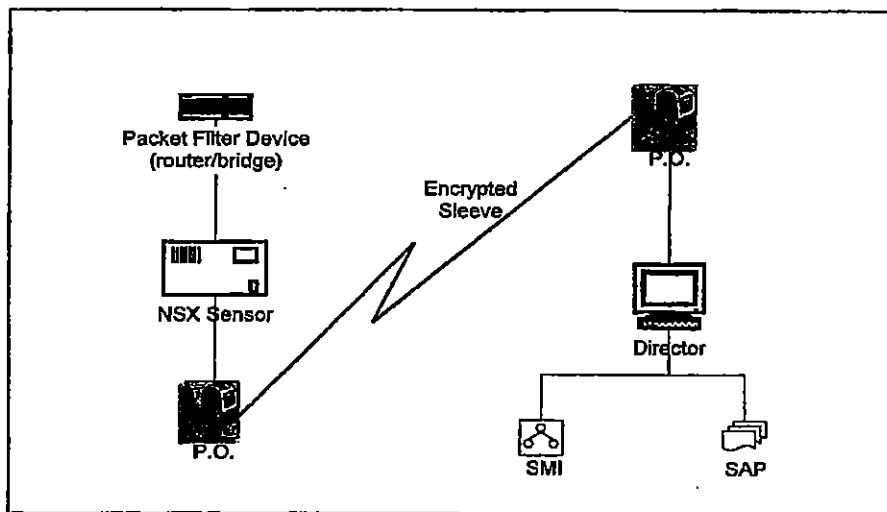


Figure 1: Basic NetRanger Components

The basic relationships between these components are described on the following pages.

The NSX

The **NSX** is the on-site filtering, sensing, and management component of the NetRanger System. It communicates with one or more remote *Director* systems via the *Post Office* network communications system. The NSX currently interfaces directly to IP networks and supports many hardware and software configuration options.

The **Packet Filtering Device** is typically a router or bridge that plugs into a network at a point of entry to other networks. The security policy installed on this device determines what subset of network traffic will be routed to the NetRanger *Sensor*.

The **Sensor** subsystem contains NetRanger's real-time intrusion detection and content assessment logic. The intrusion detection engine *recognizes* and *responds* to attacks, such as sendmail, ping sweeps, IP source routing and spoofing, FTP and telnet abuse, and SATAN scans. The Sensor's analyses produce data streams of IP packets and event records that are either dumped into a local session log files or sent on to the *Post Office* for remote delivery to a *Director* system. The Sensor also accepts intrusion response and reconfiguration information from *Director* systems.

The Communication System

The **Post Office** subsystem provides the backbone for remote monitoring and management of NSX systems. All communication is based on a proprietary connection-based protocol that can switch between alternate routes to maintain the point-to-point connections specified in its routing tables. All messages are routed based on a three-part address that includes *organization*, *host*, and *application* identifiers.

The **Encrypted Sleeve** secures transmission of data between remote protected networks. Encrypted sleeves are currently implemented via the Data Privacy Facility (DPF) that comes with Network System Corporation's (NSC) BorderGuard packet filter devices.

The Director

The Director provides the monitoring and analysis services to NetRanger, and communicates with one or more NSX systems via the communication system. The Director contains two basic subsystems: the *Security Management Interface* (SMI) and the *Security Analysis Package* (SAP).

The **SMI** is a collection of GUIs and tools that help monitor and respond to security events at one or more NSX locations. The SMI integrates with network management applications (such as HP OpenView™) via menu add-ons. While the SMI is focused primarily on real-time security event management, it also supports data analysis via the SAP.

The **SAP** is a set of data analysis tools that analyze NSX data independently of SMI activities. The SAP consists of two basic components: *database export utilities* to relational databases such as Oracle™, and one or more *data analysis tools*. Both types of components can be easily integrated into an existing SMI platform. However, WheelGroup Corporation recommends that all data be exported onto a separate host. In this way, the SMI and SAP components can be configured, secured, and fine-tuned independently.

Product Capabilities

The way NetRanger integrates many tried and true network security technologies makes it the preeminent intrusion detection system. Most network security applications fall into one of two categories: **firewalls** or **network monitoring**. Both of these technologies suffer from shortcomings not found in NetRanger.

Firewalls represent the most common form of network security, and they gained popularity in part because of their relatively simple host-based installation and friendly graphical user interfaces. The biggest problem with firewalls, however, is that they rely upon *proxy services* to enforce an organization's security policies. Proxy services erect *static* barriers that frequently block legitimate as well as illegitimate activity. This puts pressure on system administrators to open pathways, which makes them vulnerable to the very events they are supposed to guard against. Because most firewalls are host-based solutions, administration of more than one firewall at a time is difficult. The system overhead associated with proxy services also prevents most firewalls from being able to scale beyond Ethernet speeds.

Although network monitoring tools can detect unauthorized activities without having to erect barriers to entry, administrators typically have to sift through audit logs after the fact in order to find security breaches. In many instances, systems have been compromised by the time the activity has been detected. Both of these approaches also tend to only look at incoming network traffic.

NetRanger enforces an organization's security policy via **real-time response and detection of intrusive events** without having to erect static barriers. NetRanger's secure communication architecture also allows command and control, as well as system information, to be distributed across networks. This section describes these capabilities in a top-down fashion within the context of the underlying architecture.

One of the fundamental design principles of NetRanger is that the *services* required by each of the subsystems be broken apart into their atomic operational components, or *daemons*, which are diagrammed in *Figure 2*. For example, the NetRanger daemon that logs events is totally separate from the ones that perform network sensing and device management. This was done for **speed, durability, scalability, and independence**.

Each of NetRanger's daemon services is purpose-built for a specific task. This makes it possible to optimize each service without compromising the functionality of the other services. Purpose-built components also tend to be more durable than systems that manage multiple tasks, and are easier to debug and upgrade.

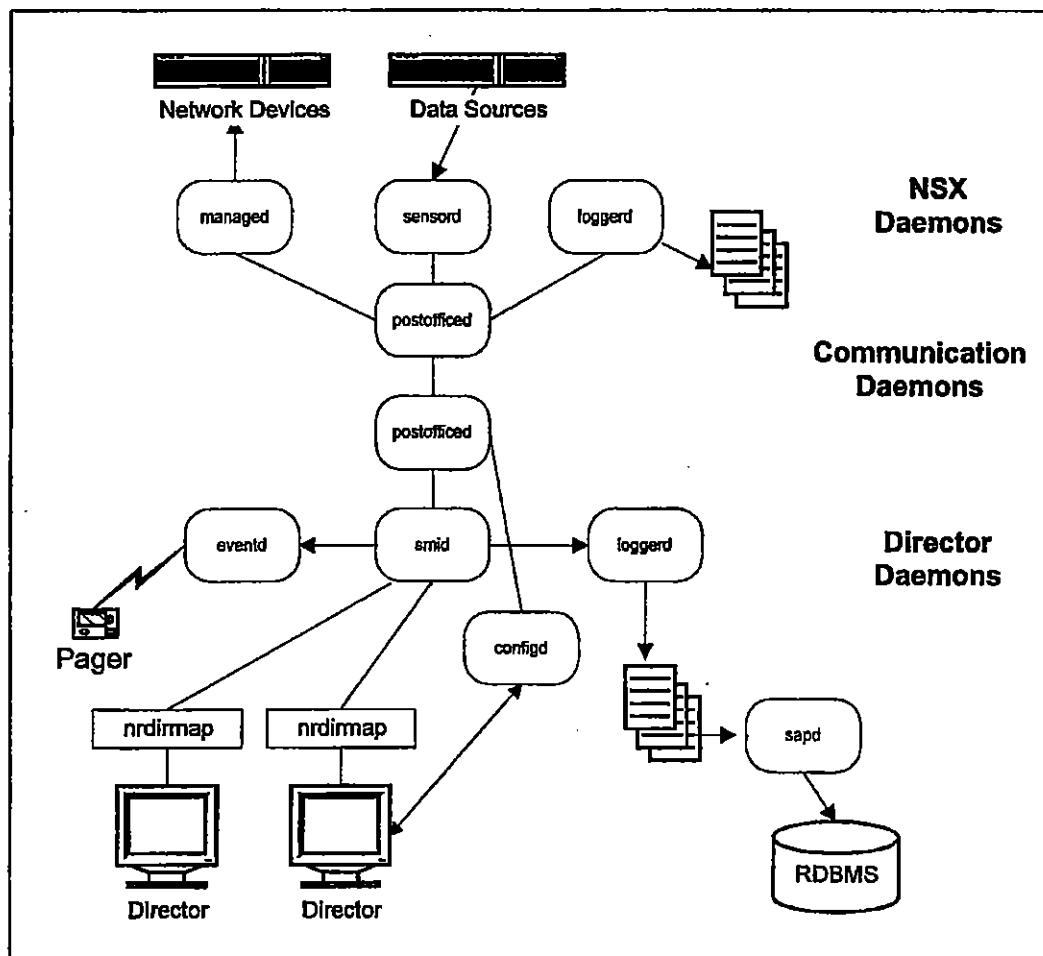


Figure 2: NetRanger 1.1 Architecture

NSX System

The NSX's capabilities are best described as **network sensing**, **device management**, and **session logging**, which have been implemented via the *sensord*, *managed*, and *loggerd* daemons diagrammed in Figure 2.

Although NetRanger is frequently described as an *Intrusion Detection* system, It also looks for a variety of suspicious activities that precede unauthorized events. An NSX will detect and report a ping sweep of a network. Although not truly intrusive, a ping sweep is frequently a precursor to unauthorized activity. Similar patterns can be found in everyday life. Many criminals drive through neighborhoods looking for homes that are vulnerable to break-ins. While not illegal, this type of activity is suspicious when it is initiated by an outsider. The NSX system looks for **network patterns of misuse** based on a variety of different **attack signatures**. Both of these topics are discussed in more detail further on.

Network Sensing

Patterns of misuse are identified by two basic types of network signatures: **context** and **content**. Context-based signatures deal with the state of a transmission as defined by the structure of packet headers, and content-based signatures focus on *what* is being transported—the binary data. As explained more fully in the *Attack Signatures* section, context-based signatures tend to be more complex than content-based ones. The steps required to identify context-based signatures are also complex and highly proprietary. As a consequence, context-based signatures are embedded within the NSX sensor subsystem, whereas content-based signatures are simple character strings that are dynamically loaded at runtime (e.g., username “root”).

Network Protocols

The sensor subsystem currently works with **TCP/IP**. Future release will support other network protocols, such as Novell’s IPX/SPX.

Packet Filter Devices

The only types of packet filter devices the sensor subsystem currently works with are the **BorderGuard** and **ERS/Passport** from NSC. These packet filter devices play a key role in the success of the NSX system.

In addition to serving as high-speed IP data sources, all of these devices

- can be **reconfigured on the fly**,
- support the same **NetSentry** interface,
- can be deployed as **bridges** as well as **routers**, and
- can maintain **Virtual Private Network (VPN)** connections.¹

Because these devices can be reconfigured on the fly, NetRanger can dynamically **shun** as well as **detect** suspicious and unauthorized network activity. The common command and control interface provided by NetSentry allows all three devices to be supported by a single set of NSX sensor and management services. It will soon be possible to operate these devices as bridges or routers, which means that an NSX can be deployed in a network behind existing devices, such as Cisco routers, without having to change routing protocols or reassign existing network addresses. And as explained more fully later on, the VPN facilities provided by NSC are the key to NetRanger’s secure communication system.

Performance Capabilities

The NSC packet filter devices fall into three distinct price/performance products: the **BorderGuard 1000**, **BorderGuard 2000**, and the **ERS/Passport**. These, in turn, serve as the basis for the three NSX configurations currently offered by WheelGroup: the NSX 1000, 2000, and 5000 systems. As with the NSC systems, the primary difference between the NSX systems is one of network performance, which is summarized in Table 1.

¹ The ERS/Passport relies upon a separate BorderGuard unit to maintain an encrypted sleeve.

	NSX 1000	NSX 2000	NSX 5000
Max Bandwidth	512 Kbps	T1 (or 10 Mbps)	T3 (or 100 Mbps)
NSC Device	BorderGuard 10000 > 2 Ethernet > 3 WAN/1 Ethernet	BorderGuard 2000 > 4 Ethernet > 2 Ethernet/2 WAN	ERS/Passport > FDDI > Ethernet > WAN > Token Ring
NSX Sensor	Pentium 166 Mhz > 2 GB Hard Drive > 32 MB RAM > 10BaseT Ethernet > modem dial-up > 2 serial ports	Pentium 166 Mhz > 2 GB Hard Drive > 32 MB RAM > 10BaseT Ethernet > modem dial-up > 2 serial ports	UltraSPARC 143 Mhz > 4 GB Hard Drive > 64 MB RAM (min) > SBUS FDDI > modem dial-up > 2 serial ports

Table 1: NSX Configurations

NetSentry

As noted at the beginning of this *Capabilities* section, NSX intrusion detection is based on the monitoring of an open network connection rather than a closed one. The NSX does this by leveraging the layered filter architecture built into NetSentry, which is diagrammed in Figure 3 below.

The *First*, *Apply* Table, and *Last* filter points apply to *all* of the network interfaces installed on a BorderGuard/ERS system, while the *Incoming* and *Outgoing* filter points allow different *In* and *out* policies to be applied to each of the network interfaces installed on the device.

One of the reasons the NSX system is able to perform *real-time* intrusion detection is because it is able to leverage the *copy_to/log_to* auditing features via filters applied to the *First* filter point. For example, the NSX's default *first.fil* filter instructs the BorderGuard/ERS to only pass on specific IP packets to *sensord*. This helps to dramatically reduce the amount of traffic the NSX system has to process.

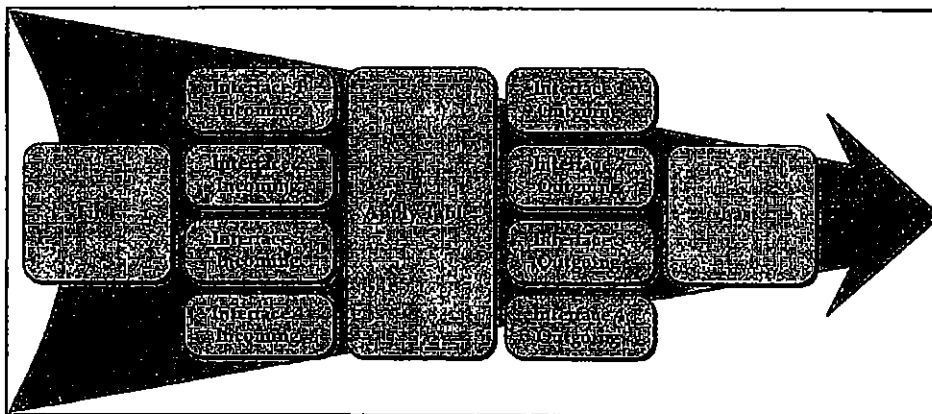


Figure 3: NetSentry PCF Filters

Attack Signatures

As previously mentioned, an attack signature is a *pattern of misuse* based on one or more events. Such a pattern can be as simple as an attempt to access a specific port on a specific host, or as complex as sequences of operations distributed across multiple hosts over an arbitrary period of time. Events can be grouped into different *attack signatures*. An event that is based on a single ICMP packet at a specific point in time is an *atomic signature*. An example of this type of signature might be a ping of a specific host. **Composite signatures**, on the other hand, are based on series of events. A ping sweep is an example of a signature that spans a network, while a port sweep is an example of a signature that focuses on a specific host. A SATAN attack is an example of a composite signature derived from a host port sweep pattern. Many of these patterns are also *stateless*, which means that the attack signature must be identified regardless of the order or the duration between atomic events. Other signatures, such as SYN attacks, are based on well-defined event sequences which are *stateful*. The full list of signatures supported by NetRanger are described in the *NetRanger User's Guide*.

Attack Responses

The NSX system is designed to do one or more of the following things once an attack has been positively identified:

- **Generate an alarm**
- **Shun the attack**
- **Log the alarm event**

In keeping with the flexibility of NetRanger, the initiation of these actions, as well as how they are initiated, is highly configurable.

Alarms

Alarms are generated by the *sensord* daemon, and typically are routed to a remote Director system. As explained more in the *Communications* section, these notifications can also be routed to *multiple* Director systems.

The type of alarm generated for a specific attack is dictated by configuration tokens stored in the NSX configuration file */usr/nr/etc/sensord.conf*. Alarm notifications are grouped according to their **severity of misuse**. NetRanger allows users to define up to 255 different levels of severity. However, the default configuration currently specifies 6 levels of misuse.

Shunning

In addition to generating alarms, *sensord* can initiate **optional actions**. The most common action is to **shun an attack**, which typically involves reconfiguration and reloading of the NetSentry *Apply Table* (as diagrammed in *Figure 3*). This type of automated response should only be configured for attack signatures with a low probability of a false positive response. A SATAN attack is an example of an unambiguous activity, whereas a content-based signature such as "vrfy" (off mail port 25) is more prone to a false positive pattern match.

Another way of shunning patterns of misuse is to manually reconfigure the BorderGuard or ERS device through the Director system. Shunning is part of a site's security policy that must be carefully reviewed before it is deployed, whether as a set of automatic rules in *sensord.conf*, or as a set of guidelines that operational staff rely upon.

Logging

All NSX log data is written to **flat files**, which can then be exported to industrial-grade databases by the SAP subsystem described in the *Director* section. Data is written directly to flat files instead of a database in order to maximize **fault tolerance** and **performance**.

The NSX system currently supports two types of logging:

- **Event logs**
- **IP Session logs**

Alarms represent just one type of *event* that can be logged by the NSX system. Event logs can also contain entries for every *command* and *error* that is generated by a user or daemon service. Data is written to these type of log files as long as NetRanger is running. Log files are **serialized** based on configurable time and size intervals defined in */usr/nr/etc/loggerd.conf*. The naming convention for Event log files is **log.<date-time>**.

IP Session logs are only written to when a certain event(s) occurs, such as a connection request from a specific IP address or a string such as "Confidential" has been detected. When these type of conditions are met, *sensord* can be configured to write every incoming and outgoing packet to an IP Session log for some predefined period of time. IP Session logs allow users to reconstruct the *conversation* that takes place between a source and destination IP addresses. The naming convention for IP Session logs is: **log.<src IP address>**.

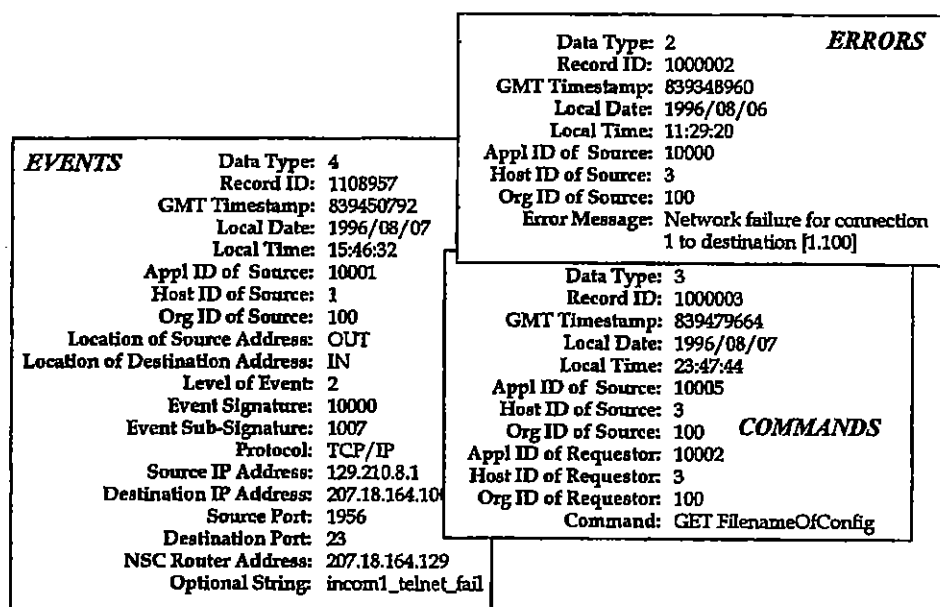


Figure 4: Event Log Formats

Timestamp	Packet Length	IP Packet
4 bytes	4 bytes	20+ bytes

Figure 5: IP Session Log Format

Note that both Event and IP Session information can be logged locally on the NSX system, as well as remotely on Director systems; exactly **what** information is sent **where** depends upon how each NSX system is configured. This is described in the next section.

Communication System

NetRanger is a **distributed application** that allows exchange of audit information and command and control across networks. All communication is based on a **proprietary, connection-based protocol** that is fault tolerant and supports alternate routes. While the communication system currently runs on top of TCP/IP networks, it has been implemented at the **application layer** of the OSI network model so that it can eventually operate on top of other protocols, such as IPX/SPX and NetBios. The underlying packet transfer mechanism employs a **sliding window** for performance and is **UDP-based** for scalability.

Communication Protocol

As Figure 6 shows, all of the NetRanger daemons communicate with one another via *postofficed daemons*. Note that this axiom holds true for communication between daemons on the same host as well as across hosts. All communication is based on a unique three-part address that includes **Organization**, **Host**, and **Application** identifiers, which are enumerated in each NSX's and Director's *organizations*, *hosts*, and *services* files in */usr/nr/etc*.

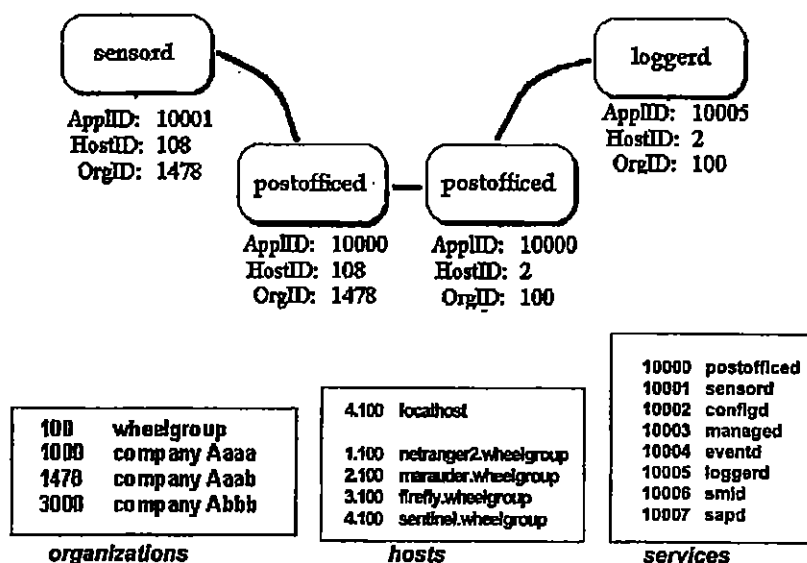


Figure 6: Example */usr/nr/etc* communication files

The benefits of this proprietary addressing scheme are twofold: 1) it can be layered on top of existing network protocols and 2) it can address a much larger domain than the current 32-bit IP protocol.

Alternate Routes

NetRanger's three-part addressing scheme serves as the basis for a point-to-point protocol that allows for up to 255 alternate routes between two hosts. These alternate pathways are defined in */usr/nr/etc/routes*, which also maps NetRanger addresses to native host addresses. *Figure 7* shows two alternate routes to the "wheelgroup" host "netranger2." The preferred path is via an IP host with the address of 207.18.164.130, while an alternate path has been identified via a host with an IP address of 10.1.4.150.

netranger2.wheelgroup	1	207.18.164.130	45000	1
netranger2.wheelgroup	2	10.1.4.150	45000	1
marauder.wheelgroup	1	10.1.4.150	45000	1

Figure 7: Example */usr/nr/etc/routes* file

An important feature of this alternate routing protocol is that it automatically switches to the next route whenever the current route fails. It also uses a system heartbeat to detect when a connection to the preferred route can be reestablished. Note that a system error message is generated (and logged) whenever a connection goes down, and any packets that were lost during a state transition are resent.

Distribution of Data

In addition to specifying alternate routes and maintaining fault tolerance, the communication protocol also allows you to define **what** types of information and **how much** information should be sent to each destination.

As noted in the section on *Logging*, the types of information generated by the NSX daemon falls into four basic categories: **Events**, **Commands**, **Errors**, and **IP Packets**. The */usr/nr/etc/destinations* file allows you to specify what types of information should be routed to which **daemons** on which **hosts**. *Figure 8* shows two different distribution entries. The first entry routes all of the standard Event data to the *loggerd* service on a Director platform named *Sentinel*, while the second entry specifies that only events should be displayed by *smid* on *firefly* Director platform.

1 sentinel.wheelgroup	loggerd	1	EVENTS,ERRORS,COMMANDS
2 firefly.wheelgroup	smid	2	EVENTS

Figure 8: Example */usr/nr/etc/destinations* file

In addition to being able to specify what *types* of information should be distributed, NetRanger can dictate what *levels* of event should be sent to each destination. As shown in *Figure 8*, only events of level '2' and above are being sent to *smid* on *firefly*.

Distribution Hierarchies

Another feature that complements alternate routing is the ability to build hierarchies of NSX and Director systems through the use of **message propagation**. Instead of broadcasting events from an NSX onto multiple hosts, information can be sent to a single host, which can then propagate

packets onto other platforms defined in its local configuration files. *Figure 9* illustrates this concept via a simple hierarchy of Director platforms. Refer to the section that describes *smid.conf* and the *DupDestination* token for more information on this feature.

In addition to providing a degree of fault tolerance, distribution hierarchies can simplify system management. For example, local Director Platforms might be responsible for monitoring from 9 am to 5 pm and then transfer control onto a central Director every evening.

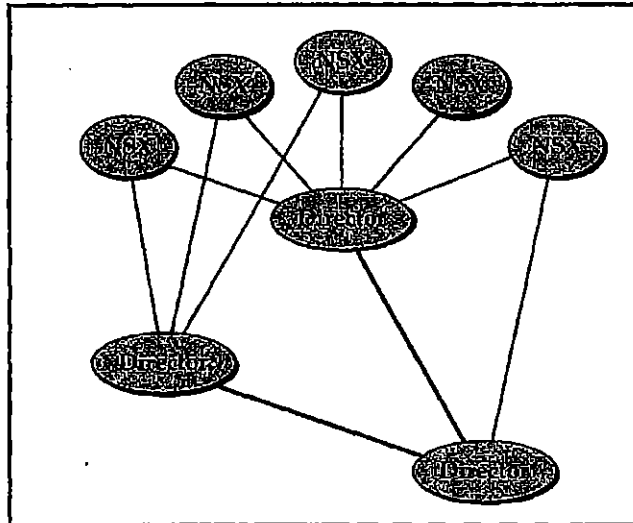


Figure 9: Director hierarchy based on message propagation

Encrypted Sleeves

A key requirement of the communication architecture is its **security**. This is achieved by passing all communication between NSX and Director systems through NSC's Data Protection Facility (DPF), which maintains Virtual Private Networks (VPN) via RSA public key and one of the following private key systems: DES, Triple DES, or IDEA. NSC's DPF is noteworthy in that:

- it can be **maintained across collections** of NSX and Director systems, and
- it is **transparent** to applications such as NetRanger

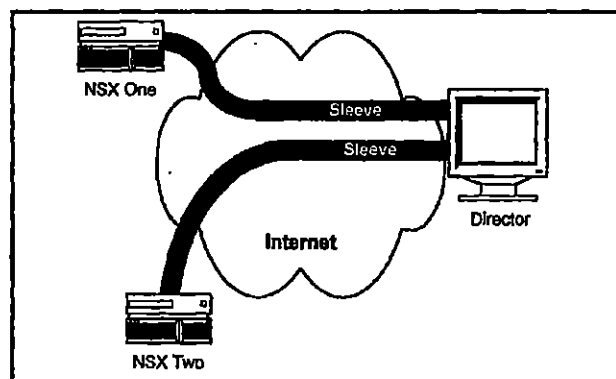


Figure 10: Example of NSC's VPN

Network Security Maps

The Director arranges icons into hierarchical security maps based on OpenView's Network Node Management (NNM) user interface. Double-clicking on an icon takes the user into the next lower "submap". Figure 12 shows several examples of *Event/Alarm* icons. The Director hierarchy includes the following levels, where *NetRanger* is the "root" and *Events/Alarms* are the "leaf" nodes of the tree:

- **NetRanger**
 - **Collections of Directors/NSXs**
 - A Single Director or NSX System
 - **Applications** running on a Director or NSX System
 - **Events/Alarms** generated by an Application



Figure 12: Sample Director Alarm Icons

Icon States

Every icon within a network security map has a *state*, which is expressed in the form of textual and graphical attributes.

The most visible indication of an icon's state is its *color*. The colors **green**, **yellow**, and **red** represent the states *normal*, *marginal*, and *critical*. These states are user-defined in an attribute dialog (shown in Figure 13); editable fields appear in gray. Level 2 and 3 alarms are usually set as *marginal* (yellow), and level 4 and 5 alarms are set as *critical* (red).

Unlike all other icon attributes, color is a state that is *inherited* from *Event/Alarm* icons by all other levels of the icon hierarchy. For example, a NSX *collection* icon is automatically set to red if any of its subordinate NSX applications has received a critical alarm. The ability to **propagate** alarm states up through a hierarchy is one of the features that makes the Director such a powerful network security monitoring tool.

Every icon in a Director hierarchy also possesses textual attributes. This information is accessed by selecting a particular icon and then choosing **Describe/Modify** from the menu bar or pressing **Ctrl+O**, which brings the icon's attributes dialog to the forefront. Note that each *type* of icon has a different set of attributes. Figure 13 shows the attributes for a *sensord* icon. Attribute dialogs for other types of icons can be found in the chapter on Operating the Director in the *NetRanger User's Guide*.

NetRanger High-Level Overview

14

Attributes for Object 100.1.10001.App:

NetRanger/Director

Application Name:
sensord

Minimum Marginal Status Severity:
2

Minimum Critical Status Severity:
4

Alarm Consolidation Threshold:
2

Organization ID:
100

Host ID:
1

Application ID:
10001

Messages:

OK Verify Cancel Help

Figure 13: *sensord* Attribute Information

NSX Management

Another key capability of the Director is remote management of the daemon processes (or *Applications*) that make up an NSX system. *Applications* are managed via a graphical user interface called *nrConfigure*, which talks directly to the *configd* daemon diagrammed in *Figure 2*. This interface allows a user to effectively *get* and *set* such attributes as log file names and alarm responses. The *nrConfigure* interface is activated by selecting one or more icons on the security map and then choosing **Security->Configure** from the OpenView menu. This opens the window shown in *Figure 14*.

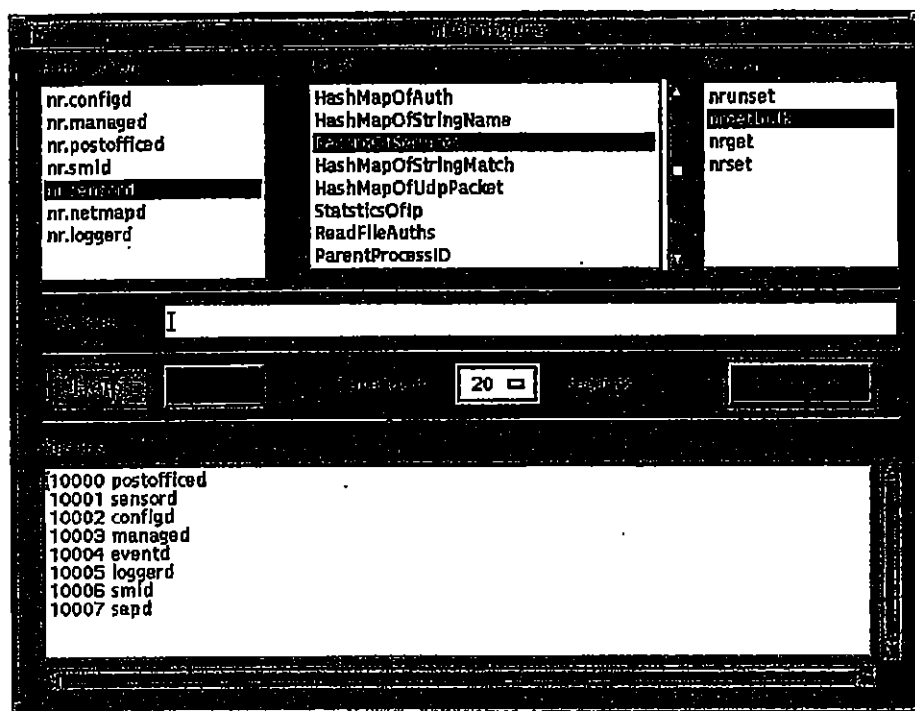


Figure 14: NSX Configuration Interface

The **Applications** displayed in the top left-hand list box represent the services running on the first NSX icon highlighted by the user. The top center list box displays all of the **Tokens** that apply to the *Application* currently highlighted in the left-hand list box. The top right-hand list box displays the **Actions** that are allowed for the currently selected *Token*. *Figure 14* shows the *Actions* and *Tokens* for the *sensord* daemon.

NetRanger currently supports over 100 tokens. An *Action* is put into effect by pressing the **Execute** button. The five possible actions are **get**, **getbulk**, **set**, **unset**, and **exec**.

The **get** and **getbulk** commands are read-only operations that obtain information from an *Application*. The **get** command returns information for a *single* item; **getbulk** returns information for a *set* of items. *Figure 14* shows the results from a **getbulk** request for all of the services defined for an NSX or Director.

The **set** command is a write operation that updates the value of a *Token*; **unset** removes a specific *Token* and its value from an Application's configuration. Adding or deleting a content-based attack signature (such as "vrfy") from an NSX is one example of these *Actions*.

The **exec** command applies to those *Tokens* which can instruct an application to execute an action external to itself. For example, a user can instruct an NSX to shun a specific IP address by issuing an **exec** against *managed*, which in turn modifies the filter configuration on its packet filter. The **exec** command is also used for such tasks as writing configuration information to disk.

It is important to understand that all configuration data is read and written to or from the *Applications* themselves, not from local caches or parameter files. The results of a **set** or **exec** take affect immediately on the target NSX.

Finally, all of the functionality provided by nrConfigure is also available via command-line interfaces. This allows trusted users without access to a Director to manage NSX systems from a simple terminal session.

NSX Data Collection

NSX data collection serves as the foundation for the SAP subsystem, and is based on the *loggerd* and *sapd* services diagrammed in Figure 15. These daemons use a simple push-pull mechanism to migrate data into a remote database. As explained earlier, *loggerd* pushes data into flat files, which are serialized based on a configurable size or time interval. This data is then pulled into a remote database by *sapd*, which has its own polling interval.

Writing to intermediate flat files in this manner provides levels of **fault tolerance** and **performance** that cannot be achieved by writing directly to a database. Data throughput in a distributed application such as NetRanger is constrained by the weakest link in the system. With the SAP system, the data capture process is insensitive to database availability or performance fluctuations.

While the SAP currently ships with *sapd* drivers only for Oracle and Remedy, it can also be configured to write to other databases, such as Sybase and Informix. Example scripts are shipped with the Director that show how a database's native bulk load tools can be easily integrated into *sapd*.

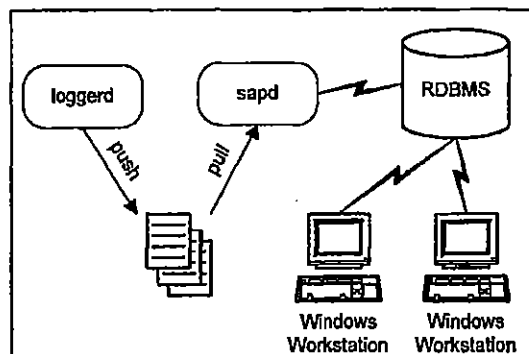


Figure 15: Push-Pull Diagram

NSX Data Analysis

The second SAP capability is data analysis. Rather than locking the user into a single tool on the Director platform, it is assumed that this task is better served by **third-party tools** on a separate Windows platform, such as the IQ Objects **report writer** from IQ Software and **multi-dimension analysis** tools such as PowerPlay from Cognos. **Trouble ticketing** systems such as Remedy's Action Request System (ARS) can also be implemented on top of NetRanger's alarm data.

These type of third-party tools can be configured to support ad hoc queries as well as predefined reports. For example, these tools can easily generate reports showing

- all alarms of levels 4 and 5 in the last 30 days,
- a graph of Web server activity over the last 24 hours, and
- a table of all events in the last 30 days in order of increasing alarms

User-Defined Actions

In addition to being able to display and log alarm events, the Director is able to generate user-defined actions via *eventd*. A typical action might be to generate pager notifications via email, or feed data onto third-party devices, such as a printer. Support for multiple action scripts is also provided. Also, *eventd* makes no distinction between alarm types and levels. However, the default action script shipped with this service shows how to trigger actions based on these criteria.

EXHIBIT D

DAY CASEBEER
MADRID & BATCHELDER LLP

20300 Stevens Creek Blvd., Suite 400
Cupertino, CA 95014
Telephone: (408) 873-0110
Facsimile: (408) 873-0220

Jonathan Loeb
(408) 342-4545
jloeb@daycasebeer.com

February 14, 2006

VIA FACSIMILE & U.S. MAIL

Katherine D. Prescott
Fish & Richardson
500 Arguello Street
Suite 500
Redwood City, CA 94063

Re: *SRI International, Inc., v. Internet Security Systems, Inc., and Symantec Corporation*

Dear Katie:

We write to provide Symantec's Rule 26(e) supplementations.

In addition to the experts identified previously, Symantec presently intends to have Dan Teal and Fred Avolio testify as expert witnesses. Symantec expects to produce documents responsive to SRI's Interrogatory No. 17 by February 20, as soon as our third-party confidentiality review and notice process has been completed. Symantec is not aware of any other discovery responses that require supplementation at this time.

Very truly yours,

DAY CASEBEER
MADRID & BATCHELDER LLP



Jonathan Loeb

JL:par

cc: Holmes Hawkins
Theresa Moehlman

EXHIBIT E

DAY CASEBEER
MADRID & BATCHELDER LLP

20300 Stevens Creek Blvd., Suite 400
Cupertino, CA 95014
Telephone: (408) 873-0110
Facsimile: (408) 873-0220

Renee DuBord Brown
(408) 342-4551
rbrown@daycasebeer.com

May 16, 2006

VIA FACSIMILE & U.S. MAIL

Michael M. Rosen
Fish & Richardson
12390 El Camino Real
San Diego, CA 92130

Re: *SRI International, Inc., v. Internet Security Systems, Inc., and Symantec Corporation*

Dear Mike:

We received your subpoena for Mr. Teal on May 12th. As an initial matter, I note that several of the categories of requested documents include materials that the parties agreed would not be subject to expert discovery. You will receive objections to this subpoena shortly.

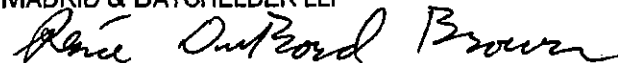
As Mr. Grewal requested in his email to you of May 15th, please confirm that Mr. Teal's deposition will begin at 8:00 a.m. on May 24th.

Given the breadth of your requests and the short time frame for response noticed in your subpoena, Mr. Teal may not be able to produce all the requested documents by May 19th. However, Mr. Teal will produce these documents by May 20th. Please let me know if the noticed Austin location can accept delivery on Saturday, May 20th, and provide me with a contact person at that location if necessary.

With regard to your specific requests, Mr. Teal is in possession of certain limited NetRanger source code files he retained for ongoing Cisco Systems consulting. However, all of this source code is the property of Cisco Systems, and Mr. Teal is not authorized to release it. Other code not subject to this restriction will be produced.

Regards,

DAY CASEBEER
MADRID & BATCHELDER LLP



Renee DuBord Brown

RDB:pw

cc: Holmes J. Hawkins III
Theresa Moehlman
Bhavana Joneja

345339

EXHIBIT F

AO 88 (Rev. 1/94) Subpoena in a Civil Case

Issued by the
UNITED STATES DISTRICT COURT

WESTERN

DISTRICT OF

TEXAS

SRI INTERNATIONAL, INC., a California
 Corporation,

Plaintiff and
 Counterclaim-Defendant,

v.

INTERNET SECURITY SYSTEMS, INC., a
 Delaware corporation, INTERNET SECURITY
 SYSTEMS, INC., a Georgia corporation, and
 SYMANTEC CORPORATION, a Delaware
 corporation,

SUBPOENA IN A CIVIL CASE

CASE NUMBER: ¹ C. A. No. 04-1199 (SLR)
 (Pending in the District of Delaware)

Defendants and
 Counterclaim-Plaintiffs.

TO: Daniel Teal

c/o Day Casebeer Madrid & Batchelder LLP, 20300 Stevens Creek Blvd., Ste. 400, Cupertino, CA 95014

☐ YOU ARE COMMANDED to appear in the United States District Court at the place, date, and time specified below
 to testify in the above case.

PLACE OF TESTIMONY	COURTROOM
	DATE AND TIME

☒ YOU ARE COMMANDED to appear at the place, date, and time specified below to testify at the taking of a
 deposition in the above case.

PLACE OF DEPOSITION Fish & Richardson P.C. One Congress Plaza; 111 Congress Avenue, Ste. 810; Austin, TX 78701	DATE AND TIME May 24, 2006 9:00 A.M.
--	--


☒ YOU ARE COMMANDED to produce and permit inspection and copying of the following documents or objects at
 the place, date and time specified below (list documents or objects):
 See Attachment A

PLACE Fish & Richardson P.C. One Congress Plaza; 111 Congress Avenue, Ste. 810; Austin, TX 78701	DATE AND TIME May 19, 2006 9:00 A.M.
--	--

☐ YOU ARE COMMANDED to permit inspection of the following premises at the date and time specified below.

PREMISES	DATE AND TIME
----------	---------------

Any organization not a party to this suit that is subpoenaed for the taking of a deposition shall designate one or more
 officers, directors, or managing agents, or other persons who consent to testify on its behalf, and may set forth, for each
 person designated, the matters on which the person will testify. Federal Rules of Civil Procedure, 30(b)(6).

ISSUING OFFICER'S SIGNATURE AND TITLE (INDICATE IF ATTORNEY FOR PLAINTIFF OR DEFENDANT)  Attorney for Plaintiff/Counterclaim Defendant SRI INTERNATIONAL, INC.	DATE May 12, 2006
---	----------------------

ISSUING OFFICER'S NAME, ADDRESS AND PHONE NUMBER John F. Horvath; Fish & Richardson P.C.; 919 N. Market Street; Suite 1100; P.O. Box 1114; Wilmington, DE 19899-1114	Telephone: 302-652-5070 Facsimile: 650-652-0607
--	--

(See Rule 45, Federal Rules of Civil Procedure, Parts C & D on the next page)

¹ If action is pending in district other than district of issuance, state district under case number.

AO 88 (Rev. 1/94) Subpoena in a Civil Case

PROOF OF SERVICE

DATE	PLACE
SERVED	
SERVED ON (PRINT NAME)	MANNER OF SERVICE
SERVED BY (PRINT NAME)	TITLE

DECLARATION OF SERVER

I declare under penalty of perjury under the laws of the United States of America that the foregoing information contained in the Proof of Service is true and correct.

Executed on _____
DATE

SIGNATURE OF SERVER

ADDRESS OF SERVER

Rule 45, Federal Rules of Civil Procedure, Parts C & D:

(c) PROTECTION OF PERSONS SUBJECT TO SUBPOENAS.

(1) A party or an attorney responsible for the issuance and service of a subpoena shall take reasonable steps to avoid imposing undue burden or expense on a person subject to that subpoena. The court on behalf of which the subpoena was issued shall enforce this duty and impose upon the party or attorney in breach of this duty an appropriate sanction, which may include, but is not limited to, lost earnings and reasonable attorney's fees.

(2) (A) A person commanded to produce and permit inspection and copying of designated books, papers, documents or tangible things, or inspection of premises need not appear in person at the place of production or inspection unless commanded to appear for deposition, hearing or trial.

(B) Subject to paragraph (d)(2) of this rule, a person commanded to produce and permit inspection and copying may, within 14 days after service of the subpoena or before the time specified for compliance if such time is less than 14 days after service, serve upon the party or attorney designated in the subpoena written objection to inspection or copying of any or all of the designated materials or of the premises. If objection is made, the party serving the subpoena shall not be entitled to inspect and copy materials or inspect the premises except pursuant to an order of the court by which the subpoena was issued. If objection has been made, the party serving the subpoena may, upon notice to the person commanded to produce, move at any time for an order to compel the production. Such an order to compel production shall protect any person who is not a party or an officer of a party from significant expense resulting from the inspection and copying commanded.

(3) (A) On timely motion, the court by which a subpoena was issued shall quash or modify the subpoena if it

(i) fails to allow reasonable time for compliance;

(ii) requires a person who is not a party or an officer of a party to travel to a place more than 100 miles from the place where that person resides, is employed or regularly transacts business in

person, except that, subject to the provisions of clause (3)(B)(iii) of this rule, such a person may in order to attend trial be commanded to travel from any such place within the state in which the trial is held, or

(iii) requires disclosure of privileged or other protected matter and no exception or waiver applies, or

(iv) subjects a person to undue burden.

(B) If a subpoena

(i) requires disclosure of a trade secret or other confidential research, development, or commercial information, or

(ii) requires disclosure of an unretained expert's opinion or information not describing specific events or occurrences in dispute and resulting from the expert's study made not at the request of any party, or

(iii) requires a person who is not a party or an officer of a party to incur substantial expense to travel more than 100 miles to attend trial, the court may, to protect a person subject to or affected by the subpoena, quash or modify the subpoena, or, if the party in whose behalf the subpoena is issued shows a substantial need for the testimony or material that cannot be otherwise met without undue hardship and assures that the person to whom the subpoena is addressed will be reasonably compensated, the court may order appearance or production only upon specified conditions.

(d) DUTIES IN RESPONDING TO SUBPOENA.

(1) A person responding to a subpoena to produce documents shall produce them as they are kept in the usual course of business or shall organize and label them to correspond with the categories in the demand.

(2) When information subject to a subpoena is withheld on a claim that it is privileged or subject to protection as trial preparation materials, the claim shall be made expressly and shall be supported by a description of the nature of the documents, communications, or things not produced that is sufficient to enable the demanding party to contest the claim.

ATTACHMENT A TO SUBPOENA OF DANIEL TEAL

DEFINITIONS AND INSTRUCTIONS

1. "You" or "your" means Daniel Teal.
2. "SRI" means SRI International, Inc., including its officers, directors, employees, agents, and attorneys.
3. "ISS" means Internet Security Systems, Inc., including its past and present officers, directors, employees, consultants, agents, and attorneys and others acting or purporting to act on its behalf, and including their predecessors, subsidiaries, parents, and affiliates.
4. "Symantec" means Symantec Corp., a Delaware corporation, including its past and present officers, directors, employees, consultants, agents, and attorneys and others acting or purporting to act on its behalf, and including their predecessors, subsidiaries, parents, and affiliates.
5. "Document" incorporates the full meaning of Federal Rule of Civil Procedure 34, and includes all tangible things, all originals (or, if originals are not available, identical copies thereof), all non-identical copies of a document, all drafts of final documents, all other written, printed, or recorded matter of any kind, and all other data compilations from which information can be obtained and translated if necessary, that are or have been in your actual or constructive custody, possession, or control, regardless of the medium on which they are produced, reproduced, or stored (including without limitation computer programs and files containing any requested information), and any recording or writing, as these terms are defined in Rule 1001, Federal Rules of Evidence, as well as any electronic documents including electronic mail, voice mail, and text messaging. Any document bearing marks, including without limitation, initials, stamped initials, comments, or notations not a part of the original text or photographic

reproduction thereof, is a separate document. Any translation of a document is a separate document.

6. "Thing" means any tangible object other than a Document.

7. "Including" shall mean "including but not limited to."

8. The terms "and" and "or" shall be construed conjunctively or disjunctively, whichever makes the individual request more inclusive.

9. The terms "refer," "referring," "relate," or "relating" as used herein include, but are not limited to the following meanings: bearing upon, concerning, constituting, discussing, describing, evidencing, identifying, concerning, mentioning, in connection with, pertaining to, respecting, regarding, responding to, or in any way factually or logically relevant to the matter described in the request.

DOCUMENTS REQUESTED

1. All documents that refer or relate to SRI, SRI's patents, SRI's products, Symantec, Symantec's products, ISS, ISS's products, and/or this litigation.
2. All documents Symantec, ISS, and/or their attorneys have provided to you.
3. All documents you have provided to Symantec, ISS, and/or their attorneys.
4. All documents considered in preparation of your expert reports and declarations in this matter.
5. Invoices or records that refer or relate to your work for Symantec and/or ISS in this matter, including but not limited to invoices submitted to Symantec and/or ISS and documents that refer or relate to payments by Symantec and/or ISS to you.
6. All documents, including issued patents, patent applications, and articles, authored or co-authored by you relating to this litigation.
7. All Documents concerning NetRanger as defined in paragraph 12 of the Expert Report of Daniel Teal.
8. All Documents concerning WheelGroup as defined in paragraph 4 of the Expert Report of Daniel Teal.
9. All press releases, presentations, User Guides and Manuals, and other documentation developed by WheelGroup and others contained in your personal files as described in paragraph 13 of the Expert Report of Daniel Teal.
10. All Documents contained in your "detailed files of product descriptions and documentation" as described in paragraph 13 of the Expert Report of Daniel Teal.

11. All Documents Concerning any conversations that you have had within the past two years with Scott Olson, Scott Waddell, Kevin Wiley, or Jerry Lathem regarding WheelGroup and/or NetRanger.

CERTIFICATE OF SERVICE

I hereby certify that on this 12th day of May 2006, I served a copy of the **SRI INTERNATIONAL, INC.'S NOTICE OF DEPOSITION OF DANIEL TEAL** on the following individuals in the manner indicated:

VIA CM/ECF AND HAND DELIVERY

Richard L. Horwitz
David E. Moore
Potter Anderson & Corroon LLP
Hercules Plaza
1313 North Market Street, 6th Floor
P.O. Box 951
Wilmington, DE 19899
Telephone: 302-984-6000
Facsimile: 302-658-1192
Email: rhorwitz@potteranderson.com
Email: dmoore@potteranderson.com

Attorneys for
Defendant/Counterclaim Plaintiffs
Internet Security Systems, Inc., a
Delaware corporation, and Internet
Security Systems, Inc., a Georgia
corporation

VIA CM/ECF AND HAND DELIVERY

Richard K. Herrmann
Morris James Hitchens & Williams LLP
222 Delaware Avenue, 10th Floor
P.O. Box 2306
Wilmington, DE 19899-2306
Telephone: 302-888-6800
Facsimile: 302-571-1750
Email: rherrmann@morrisjames.com

Attorneys for
Defendant/Counterclaim Plaintiff
Symantec Corporation

VIA ELECTRONIC AND U. S. MAIL

Holmes J. Hawkins, III
Natasha H. Moffitt
King & Spalding LLP
1180 Peachtree Street
Atlanta, GA 30309
Telephone: 404-572-4600
Facsimile: 404-572-5145
Email: hhawkins@kslaw.com
Email: nmoffitt@kslaw.com

Attorneys for
Defendant/Counterclaim Plaintiffs
Internet Security Systems, Inc., a
Delaware corporation, and Internet
Security Systems, Inc., a Georgia
corporation

VIA ELECTRONIC AND U. S. MAIL

Theresa A. Moehlman
Bhavana Joneja
King & Spalding LLP
1185 Avenue of the Americas
New York, NY 10036
Telephone: 212-556-2100
Facsimile: 212-556-2222
Email: tmoehlman@kslaw.com
Email: bjoneja@kslaw.com

Attorneys for
Defendant/Counterclaim Plaintiffs
Internet Security Systems, Inc., a
Delaware Corporation, and Internet
Security Systems, Inc., a Georgia
Corporation

VIA ELECTRONIC AND U. S. MAIL

Lloyd R. Day, Jr.
Robert M. Galvin
Paul S. Grewal
Day Casebeer Madrid & Batchelder, LLP
20300 Stevens Creek Boulevard, Suite 400
Cupertino, California 95014
Telephone: 408-873-0110
Facsimile: 408-873-0220
Email: pgrewal@daycasebeer.com

Attorneys for
Defendant/Counterclaim Plaintiff
Symantec Corporation

/s/ John F. Horvath

John F. Horvath

10628007.DOC

EXHIBIT G

DAY CASEBEER
MADRID & BATCHELDER LLP

20300 Stevens Creek Blvd., Suite 400
Cupertino, CA 95014
Telephone: (408) 873-0110
Facsimile: (408) 873-0220

Paul S. Grewal
(408) 342-4543
pgrewal@daycasebeer.com

May 19, 2006

VIA OVERNIGHT COURIER SERVICE

Michael M. Rosen
Jason W. Wolff
Fish & Richardson PC
12390 El Camino Real
San Diego, CA 92130

Katherine D. Prescott
Fish & Richardson PC
500 Arguello Street, Suite 500
Redwood City, CA 94063

Re: *SRI International, Inc., v. Internet Security Systems, Inc., and Symantec Corporation*

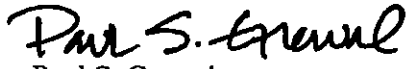

Dear Michael:

Enclosed please find a total of four CDs containing documents responsive to the subpoena issued to Daniel Teal on May 12, 2006.

Please contact me if you have any questions.

Best regards,

DAY CASEBEER
MADRID & BATCHELDER LLP


Paul S. Grewal
PSG:vnn 

Encl.: Production CDs / SYMC-Teal Production Log